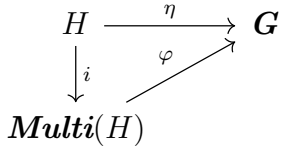
Definición 7. Sea H una signatura de multicategoría y sea $a \in H_0$, definimos recursivamente un **árbol con raíz** a **de** H, como:

- Un nodo de la forma $w : \varepsilon \to a$ es un árbol (a los nodos de esta forma, les llamamos hojas);
- Un par $(f, (g_1, \ldots, g_n))$ donde f es un nodo de aridad n, $f : a_1 a_2 \cdots a_n \to a$, y $g_1, \ldots, g_n$ son nodos con codominios $a_1, \ldots a_n$ es un arbol;
- Son todos.

Sea H una signatura de multicategoría. **Multi**(H) es la multicategoría cuyos objetos son exactamente los mismos que H con los morfismos los árboles en H y para cada objeto $a \in G_0$ tenemos la $\mathtt{Id}_a$. La $i-$ésima composición parcial para morfismos f, g en **Multi**(H), $g \circ_i f$ está por reemplazar la $i-$ésima hoja de f por una copia de g.
Decimos que dos árboles en **Multi**(H) son iguales si puede ser obtenido uno a partir de otro mediante la aplicación consecutiva de las propiedades 1 y 2 para la composición en multicategorías.
De acuerdo presentado previamente, la composición parcial anterior induce correctamente la composición y, por lo tanto, **Multi**(H) efectivamente es un multicategoría. Más aún, la construcción anterior es libre en virtud de la siguiente proposición.

Proposición 4. *Sean H una signatura de multicategorías, $\boldsymbol{G}$ un multicategoría y $f : G_1 \to O_1$ una función que preserve la aridad de los nodos. Entonces existe un único morfismo de multicategorías $\varphi :$ **Multi**$(H) \to \boldsymbol{G}$ tal que el siguiente diagrama conmuta:*



3.2. Lenguajes libres de contexto

Sea G una signatura finita de multicategoría. Consideremos sus objetos como símbolos, los nodos como reglas de producción y los morfismos en **Multi**(G) como derivaciones, entonces obtenemos la noción de una gramática, más precisamente:

Definición 8. Un **gramática de multicategorías** es una signatura finita de multicategoría de la siguiente forma:

$$(B + V)^* \xleftarrow{\mathtt{dom}} G \xrightarrow{\mathtt{cod}} B$$

donde V es un vocabulario y B es un conjunto de símbolos no terminales con un símbolo distinguido $s \in B$ y G un conjunto de reglas de producciones.
El lenguaje generado por G es:

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \in \mathbf{Multi}(G)\}$$

Para comprender la definición anterior, consideremos los siguientes ejemplos.

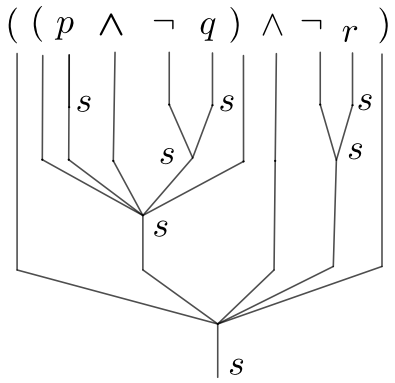
Ejemplo 6. El lenguaje de la lógica proposicional. Sea el conjunto de símbolos terminales vacío, el vocabulario $V = Var \cup \{\wedge, \neg, (,)\}$ donde Var es el conjunto de letras proposicionales con las asignaciones:

$$\text{para todo } p \in Var, \qquad p \to s$$

y las reglas de producción

$$(s \wedge s) \to s \qquad\qquad \neg s \to s$$

Con lo anterior, podemos obtener, por ejemplo, la siguiente derivación



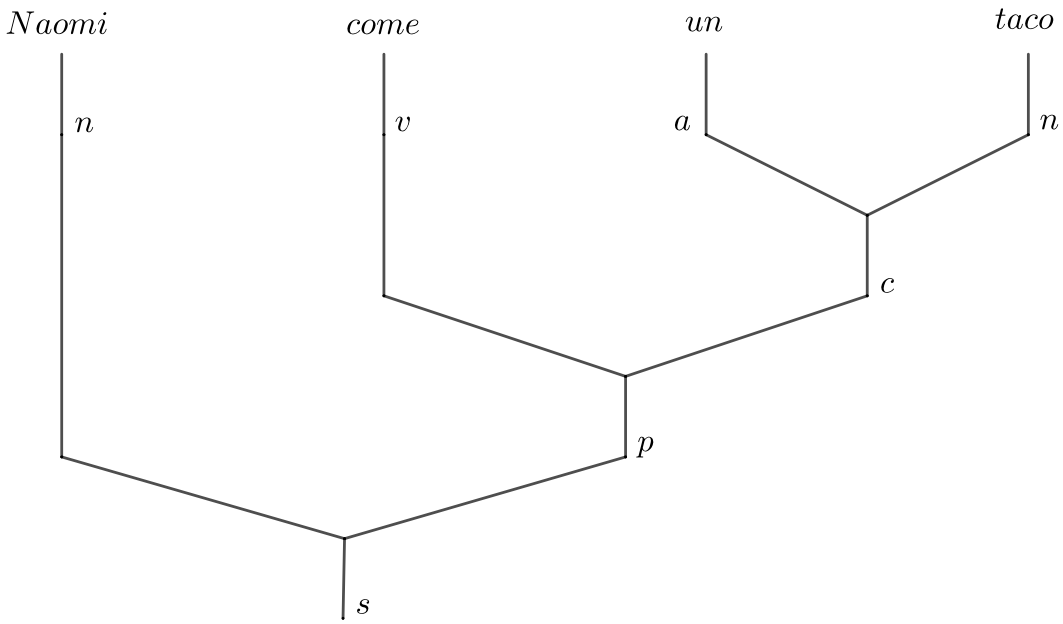
Ejemplo 7. Consideremos el siguiente conjunto de símbolos $B = \{n, v, a, c, p\}$ que corresponde a sustantivo, verbo, artículo, complemento y predicado; el vocabulario $V = \{Naomi, come, un, taco\}$; las asignaciones

$$Naomi \to n \qquad come \to v \qquad un \to n \qquad taco \to n$$

y las reglas de producción

$$(a, n) \to c \qquad (v, c) \to p \qquad (n, p) \to s$$

Tenemos la siguiente derivación



Ahora, como anunciamos al inicio del capítulo, mostraremos la relación entre los lenguajes anteriores con la jerarquía de Chomsky. Para ello, veamos las siguientes definiciones y resultados.

Definición 9. Una gramática libre de contexto es una tupla $G = (V, B, P, s)$ donde V es un vocabulario, B es un conjunto de símbolos no terminales con un símbolo distinguido s y P es un conjunto de reglas de producción de la forma $A \to \alpha$ donde A es un símbolo no terminal y $\alpha \in (V + B)^*$.
Si $A \to \alpha$ es una regla de producción, para cualquier $\beta, \gamma \in (V + B)^*$ definimos la derivación $\beta A\gamma \Rightarrow \beta\alpha\gamma$. Consideramos $\Rightarrow^*$ como la cerradura transitiva de $\Rightarrow$.
Definimos el lenguaje generado por G como:

$$\mathcal{L}(G) = \{w \in V^* | \exists s \Rightarrow w\}$$

Decimos que un lenguaje L es libre de contexto si existe una gramática G libre de contexto tal que $L = L(G)$

Observemos que una oración en un lenguaje libre de contexto corresponde a una cadena en V^* que puede ser derivada a partir de s en G.
Ahora, la manera anterior de describir el lenguaje es eficiente para la generación de nuevas expresiones, sin embargo, puede no ser muy útil para el propósito de discriminar si una expresión dada pertenece, o no, al lenguaje. Con ese propósito surgen los árboles de derivación.

Definición 10. Sea $G = (V, B, P, s)$ una gramática libre de contexto.
Un árbol es un árbol de derivación para G si:

- Cada vértice está etiquetado con un símbolo de $V \cup B \cup \{\varepsilon\}$;

- la raíz del árbol es s;
- si un vértice interior está etiquetado con A, entonces $A \in B$, es decir, no es un símbolo terminal;
- si un vértice A tiene n hijos, $A_1, \ldots, A_n$, entonces:

$$A \to A_1 A_2 \cdots A_n$$

  es una regla de producción en P; y
- si un vértice está etiquetado con ε, entonces es un hoja y, además, es el único hijo de su padre.

Notemos que el diagrama del ejemplo uno corresponde a un árbol de derivación en la gramática $A \to \neg A | A \wedge A | A \vee A | A \to A | A \leftrightarrow A | a_0 | a_1 | a_2$.
Con esto, podemos enunciar la siguiente proposición cuya prueba podemos consultar en [2].

Proposición 5. *Sea $G = (V, B, P, s)$ una gramática libre de contexto. Entonces $s \Rightarrow^* w$ si y solo si existe un árbol de derivación para G que produce a w.*

Dada la definición de morfismos en una multicategoría libre, es inmediato notar que los morfismos de la forma $f : u \to s$ con $u \in V^*$ en una gramática de multicategoría son árboles de derivaciones y viceversa. Así, tenemos el siguiente teorema con el cual terminamos el presente capítulo.

Teorema 2. *Sea G una signatura de multicategoría y G' una gramática libre de contexto, entonces*
1. $L(G)$ es un lenguaje libre de contexto.
2. Existe una signatura de multigráfica que genera a $L(G')$.

4 Gramáticas recursivamente enumerables

La introducción de los lenguajes libres de contexto aumenta ampliamente la clase de lenguajes que podemos describir formalmente; sin embargo, consideremos las siguientes oraciones presentadas en [5]: "Ayer una mujer llegó que conocía", "Ayer un mujer y un hombre llegaron a los que cononocía y admiraba". Notemos que la concordancia entre los sujetos "hombre y mujerr' y el verbo "llegaron" y el pronombre relativo 'los que' marca la estructura de oración $xa^ib^ic^id^i$, el cual no es un lenguaje regular.
En este capítulo daremos por terminado el estudio de la jerarquía de Chomsky por medio de la teoría de categorías, presentando los lenguajes recursivamente enumerables que permiten representar oraciones como las anteriores, los cuales son comúnmente definidos de la siguiente forma:

- son los lenguajes producidos por una gramática de tipo 0, las cuales son las gramáticas generativas cuyas reglas de producción tienen cero restricciones; y
- son los lenguajes reconocidos por una máquina de Turing

En esta ocasión, las categorías adecuadas para trabajar los lenguajes recursivamente enumerables son las categorías monoidales que, como su nombre lo indica, son categorías dotadas de una operación binaria que cumple propiedades análogas a la estructura de un monoide algebraico. Además, tienen asociado un cálculo gráfico, es decir, existen diagramas que nos permiten representar los morfismos en la categoría y, más aún, nos permiten efectuar operaciones en ellos mediante deformaciones. Estos diagramas nos permitirán definir las categorías monoidales libres de forma combinatoria.
Finalmente, mostraremos que los lenguajes generados por una categoría monoidal son exactamente los lenguajes recursivamente enumerables.

4.1. Categorías monoidales

Definición 11. Una **signatura monoidal** G es una signatura de la siguiente forma:

$$G_0^* \xleftarrow{\texttt{dom}} G_1 \xrightarrow{\texttt{cod}} G_0^*$$

Un **morfismo de signaturas monoidales** $\varphi : G \to \Gamma$ es un par de funciones $\varphi_0 : G_0 \to \Gamma_0$ y $\varphi_1 : G_1 \to \Gamma_1$ tales que el siguiente diagrama conmuta:

$$\begin{array}{ccccc} G_0^* & \xleftarrow[\texttt{cod}]{} & G_1 & \xrightarrow{\texttt{dom}} & G_0^* \\ \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_1} & & \downarrow{\scriptstyle \varphi_0} \\ \Gamma_0^* & \xleftarrow[\texttt{cod}]{} & \Gamma_1 & \xrightarrow{\texttt{dom}} & \Gamma_0^* \end{array}$$

Con tales morfismos, las signaturas monoidales forman la categoría **MonSig**.

Definición 12. Una **categoría monoidal estricta** $(\mathcal{C}, \otimes, 1)$ es una categoría $\mathcal{C}$ con un bifuntor[1] $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ y un objeto distinguido 1 en $\mathcal{C}$ en la que se satisfacen las siguientes ecuaciones:

1. $1 \otimes f = f = f \otimes 1$

2. $f \otimes (g \otimes h) = (f \otimes g) \otimes h$

para cualesquiera f, g, h morfismos en $\mathcal{C}$.
Un funtor monoidal estricto es un funtor entre categorías monoidales $F : \mathcal{C} \to \mathcal{D}$ que preserva el producto tensorial, es decir, $F(g \otimes_{\mathcal{C}} f) = F(g) \otimes_{\mathcal{D}} F(g)$. A la categoría de categorías monoidales con morfismos los funtores monoidales la denotamos **MonCat**

Notemos que en una categoría monoidal nuestros morfismos pueden componerse de dos maneras. Dado $f : A \to B$ y $g : B \to C$ morfismos podemos construir la composición $g \circ f : A \to C$. Pero también dados $u : A \to C$ y $v : B \to D$ podemos usar nuestro funtor para formar el morfismo $u \otimes v : A \otimes B \to C \otimes D$.
Es usual interpretar ambas composiciones de la siguiente manera: la composición usual es secuencial, es decir, primero se ejecuta el primer morfismo y después aplicamos el segundo; y la composición monoidal es paralela, es decir, ejecutamos ambos morfismos de manera simultánea.
Las consideraciones anteriores motivan un lenguaje gráfico para categorías monoidales mediante cables y cajas. Definimos recursivamente el lenguaje gráfico de la siguiente manera: Sean A, B, C, D objetos y $f : A \to B$, $g : B \to C$ y $h : C \to D$ morfismos en una categoría monoidal, tenemos la siguiente representación gráfica. Estos diagramas son llamados **diagramas de cables**.

[1] Un bifuntor es un funtor en cada entrada

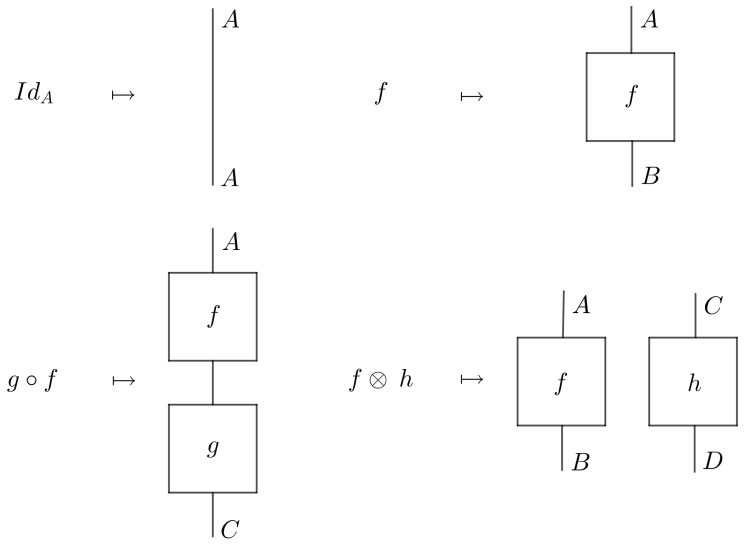


Figura 4.1: Lenguaje gráfico para categorías monoidales

Como es de esperarse, las dos operaciones entre los morfismos de nuestra categorías se "comportan bien" entre sí, como veremos en la siguiente proposición.

Proposición 6. *Sea $(\mathcal{C}, \otimes, 1)$ una categoría monoidal y sean $f : A \to B$, $g : B \to C$, $h : D \to E$ y $j : E \to F$ morfismos de $\mathcal{C}$, entonces*

$$(g \circ f) \otimes (j \circ h) = (g \otimes j) \circ (f \otimes h) \tag{4.1}$$

$$(Id_B \otimes h) \circ (f \otimes Id_D) = f \otimes h = (f \otimes Id_E) \circ (Id_A \otimes h) \tag{4.2}$$

Demostración. Sean $f : A \to B$, $g : B \to C$, $h : D \to E$ y $j : E \to F$ morfismos en una categoría monoidal $\mathcal{C}$.
Primero probemos 4.1.

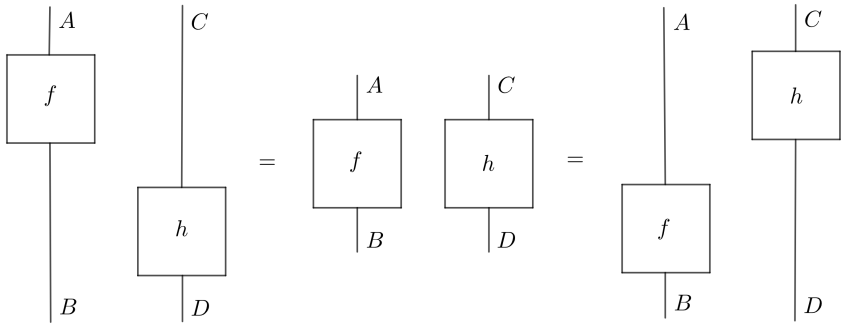
$$\begin{aligned}(g \circ f) \otimes (j \circ h) &= \otimes(g \circ f, j \circ h) \\ &= \otimes((g, j)) \circ (f, h)) \\ &= (\otimes(g, j)) \circ (\otimes(f, h)) \qquad \text{por la bifuntorialidad de } \otimes \\ &= (g \otimes j) \circ (f \otimes h)\end{aligned}$$

Ahora, veamos 4.2

$$\begin{aligned}(Id_B \otimes h) \circ (f \otimes Id_D) &= (Id_B \circ f) \otimes (h \circ Id_D) \qquad \text{Por 4.1} \\ &= f \otimes h \\ &= (f \circ Id_A) \otimes (Id_E \circ h) \\ &= (f \otimes Id_E) \circ (Id_A \otimes h) \qquad \text{Por 4.1}\end{aligned}$$

□

Podemos expresar en el lenguaje gráfico de categorías monoidales la ecuación 4.2 de la siguiente forma:

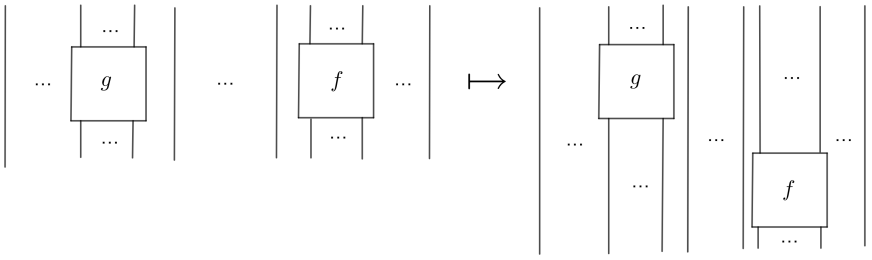


La ecuación anterior sugiere que siempre podemos encontrar diagramas donde cada caja este en un nivel distinto al resto, así que probémoslo, pero antes veamos la siguiente definición para poder enunciar nuestra proposición.

Definición 13. Un diagrama de cables está en **posición general** si no hay otra caja a la derecha o a la izquierda de cada caja.

Proposición 7. *Todo diagrama de cables es equivalente a un diagrama de cables en posición general.*

Demostración. Sea D un diagrama de cables. Haremos la demostración por inducción sobre el número de cajas en el diagrama.
La base de inducción es trivial, pues si D tiene una caja ya acabamos.
Supongamos que el resultado es cierto para algún n y que D tiene $n+1$ cajas.
Sea f la caja ubicada más a la derecha y abajo en el diagrama y consideremos D' el diagrama obtenido al sustituir f por $id_{dom(f)}$. Entonces D' tiene n cajas y por hipótesis es equivalente a un diagrama de cajas G' en posición general; consideremos ahora el diagrama G que se obtiene al pegarle f en el cable al que se lo habíamos cortado.
Observemos que la caja que colocamos sigue siendo la más derecha y abajo, y a su izquierda hay a lo más una caja g (si no, ya acabamos), pues G' está en posición general. Entonces basta hacer la siguiente deformación o, en otras palabras, aplicar 4.1.

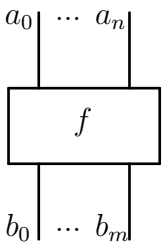


□

Notemos que el argumento de la prueba anterior fue expresado de dos formas distintas, por medio de la igualdad 4.1 y por su representación en el diagrama de cables. Sin embargo, es válido si el lector considera que fue una coincidencia muy afortunada y tenga sus reservas. Sin embargo, el siguiente teorema debido a A. Joyal y R. Street nos iluminará en el uso de diagramas de cables, cuya demostración puede consultarse en [6]

Teorema 3 (Teorema de coherencia del cálculo gráfico en categorías monoidales)**.** *Una ecuación $f = g$ bien formada[2] entre morfismos de una categoría monoidal es válida en la categoría monoidal si y sólo si los diagramas de cables de f y g son equivalentes salvo isotopías planas.*
Dos diagramas de cables son equivalentes salvo isotopías planas si es posible transformar uno en el otro moviendo continuamente las cajas y cables sin permitir que estos se crucen entre sí.

La proposición y teorema anterior serán fundamentales en nuestro objetivo: generar una categoría monoidal libre a partir de una signatura monoidal. Para ello, primero describiremos un lenguaje gráfico para las signaturas monoidales ya que lo usaremos en la descripción de categorías monoidales.
Sea G una signatura monoidal y sea $f : \vec{a} \to \vec{b} \in G_1$. Diremos que f es una caja y la representaremos como:



Además, tenemos dos casos distinguidos que corresponden a morfismos de la forma $u : \varepsilon \to a$ y $w : b \to \varepsilon$ cuya representación es la siguiente.

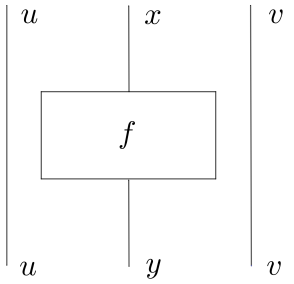
Usaremos la descripción anterior para construir la categoría monoidal libre $\mathbf{MC}(G)$ asociada a una signatura monoidal; para ello, primero veamos qué es una signatura

[2]Una ecuación entre morfismos es bien formada si los morfismos efectivamente son parte la categoría y poseen tanto el mismo dominio como codominio

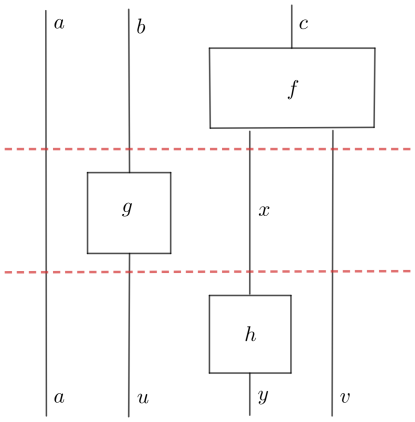
de niveles.
Dada G una signatura monoidal definimos la **signatura de niveles** de G como

$$G_0^* \xleftarrow[\texttt{cod}]{} L(G) = G_0^* \times G_1 \times G_0^* \xrightarrow{\texttt{dom}} G_0^*$$

donde para cada nivel $l = (u, f : x \to y, v)$ en $L(G)$ definimos $\texttt{dom}(l) = uxv$ y $\texttt{cod}(l) = uyv$, gráficamente representamos a l de la siguiente manera

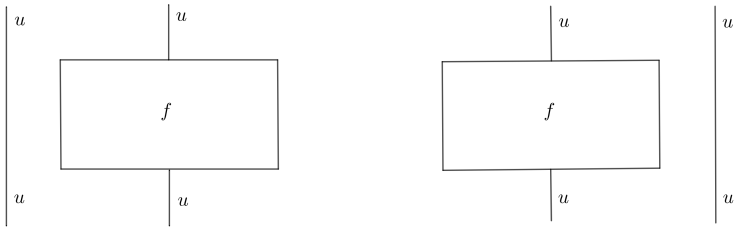


Sea $\mathbf{PMC}(G)$, la categoría libre cuyos objetos son los diagramas premonoidales y el conjunto de morfismos generadores son secuencias de niveles de la forma $(d_1, \ldots, d_n)$ tales que $\texttt{cod}(d_i) = \texttt{dom}(d_{i+1})$ para cada $i \in \{1, ..., n-1\}$.
Veamos el siguiente ejemplo de un diagrama premonoidal generado con la signatura $G_0 = \{a, b, c, u, v, y, x\}$ y $G_1 = \{f : c \to xv, g : b \to u, h : x \to y\}$



Supongamos que queremos construir el diagrama a partir de la signatura, ¿qué información necesitamos? En principio, conocer el número de niveles que conforma el diagrama, en nuestro caso tres, luego conocer el dominio y codominio del diagrama, abc y $auyv$ respectivamente, además de la lista de cajas en cada nivel, $[f, g, h]$. Finalmente, necesitamos conocer la posición en la que se encuentra la caja en cada nivel, lo cual lo identificamos señalando el número de cables a la izquierda de la caja, $(2, 1, 2)$. Aunque la última condición es innecesaria en el diagrama anterior, es importante en casos como el siguiente, donde los primeros datos no determinan un único diagrama

premonoidal.



La explicación anterior motiva nuestra siguiente proposición, una manera eficiente de representar la información contenida en un diagrama premonoidal:

Proposición 8 (Codificación de diagramas premonoidales)**.** *Sea G una signatura monoidal. Un diagrama premonoidal d en $\boldsymbol{PMC}(G)$ está únicamente determinada de la siguiente forma:*

1. *un dominio:* $\texttt{dom}(d) \in G_0^*$
2. *un codominio:* $\texttt{cod}(d) \in G_0^*$
3. *una lista de cajas:* $\texttt{cajas}(d) \in G_1^n$
4. *una lista de números identificadores:* $\texttt{id}(d) \in \mathbb{N}^n$

donde n es el número de niveles del diagrama d y el $n-$ésimo identificador señala la posición de la caja en el $n-$ésimo nivel, indicando el número de "cables" a la izquierda de la caja.
Más aún, la información anterior define un diagrama premonoidal válido si para cada $\imath \in \{1, ..., n\}$ tenemos que

$$\texttt{peso}(d)_i \geq \texttt{id}(d)_i + |\texttt{dom}(b_i)|$$

donde

$$\texttt{peso}(d)_1 = |\texttt{dom}(d)| \qquad \texttt{peso}(d)_{i+1} = \texttt{peso}(d)_i + |\texttt{cod}(b_i)| - |\texttt{dom}(b_i)|$$

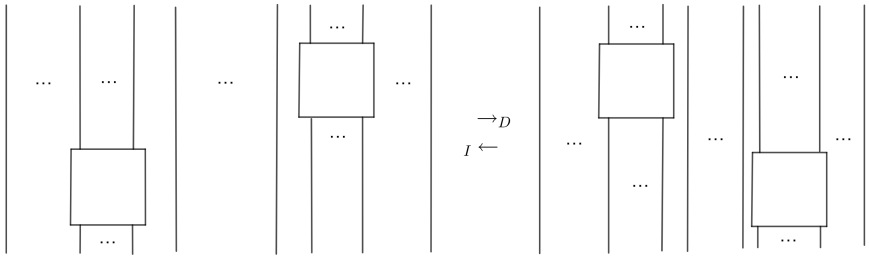
con $b_i = \texttt{boxes}(d)_i$

Notemos que la función `peso` indica el número de cables con los que inicia cada nivel; así, la desigualdad significa que este número debe ser mayor que el número de cables que estarán a la izquierda de la caja menos el número de cables que necesita la caja del nivel. Bajo estas condiciones, es evidente que el diagrama generado será válido.
Ahora bien, notemos que los elementos de $\mathbf{PMC}(G)$ se parecen mucho a diagramas de cables en posición general; sin embargo, necesitamos establecer cuándo dos diagramas premonoidales son equivalentes. Para ello, veamos las siguientes definiciones.

Definición 14. Sea G una signatura y d un diagrama premonoidal de G. Decimos que d admite un intercambio izquierdo en el nivel n si $\texttt{id}(d)_{n+1} \geq \texttt{id}(d)_n + |cod(b_n)|$ y admite un intercambio derecho en el nivel $n+1$ si $\texttt{id}(d)_n \geq \texttt{id}(d)_{n+1} + |dom(b_n)|$.

En otras palabras, un intercambio es posible cuando las cajas a intercambiar no tienen cables en común. Ahora sí, veamos qué es un intercambio.

Definición 15. Sea G una signatura y d un diagrama premonoidal de G. Si d admite un intercambio izquierdo (o derecho) en el nivel n, entonces generamos un nuevo diagrama idéntico en todos los niveles a d salvo en n y $n+1$ donde hay un cambio de la siguiente forma:



Decimos que el nuevo diagrama es el **intercambio izquierdo (o derecho) en el nivel** n **de** d.

Claramente los diagramas producidos tras un intercambio son también un diagrama válido. Definimos una reducción de diagramas como una secuencia de intercambios izquierdos o derechos. Con ello, definimos la siguiente relación: dos diagramas d y d' están $\sim$-relacionados si existe una reducción que lleve d a d'.

Proposición 9. *Sea G una signatura monoidal. La relación $\sim$ sobre el conjunto de diagramas premonoidales de G es de equivalencia.*

Demostración. Sea G una signatura monoidal. Sea d una diagrama premonoidal. Consideramos la secuencia de reducciones vacía, entonces $d \sim d$. Por lo tanto, $\sim$ es reflexiva.
Sea d y d' diagramas tales que $d \sim d'$. Sea $p_1^{r_1}, p_2^{r_2}, ..., p_n^{r_n}$ la secuencia de reducciones que lleva d a d' con $r_i \in \{d, i\}$ para todo $i \in \{1, ..., n\}$, es decir, indica si el intercambio fue izquierdo o derecho. Notemos que un intercambio izquierdo es el inverso del derecho y viceversa, entonces consideremos r_i' como el opuesto a r_i. Entonces la reducción $p_n^{r_n'}, p_{n-1}^{r_{n-1}'}, ..., p_1^{r_1'}$ lleva d' a d. Por lo tanto, $d' \sim d$, así tenemos que $\sim$ es simétrica.
Por último, sean d, d' y d'' tales que $d \sim d'$ y $d' \sim d''$. Entonces hay una reducción que lleva d a d' y una que lleva d' a d'', así hay una reducción que lleva d a d'', es decir, $d \sim d''$. Por lo tanto, $\sim$ es transitiva.
Por lo tanto, $\sim$ es de equivalencia. □

Otra forma de expresar la proposición anterior es señalando que cualesquiera dos diagramas premonoidales tales que pueda deformarse, sin cruzar o cortar cables, uno en el otro están $\sim$-relacionados. Con todo lo anterior, ya estamos en condiciones de conocer a nuestra categoría monoidal libre, $\mathbf{MC}(G)$.

Proposición 10. *Sea G una signatura monoidal, entonces definimos la categoría monoidal libre de G como*

$$\boldsymbol{MC}(G) \cong \boldsymbol{PMC}(G)/\sim$$

donde $\boldsymbol{PMC}(G)/\sim$ es la categoría premonoidal libre con la identificación en morfismos: si $f \sim g$, entonces $f = g$.

Demostración. Sea G una signatura monoidal. Mostremos que $\mathbf{PMG}(G)/\sim$ es una categoría monoidal.
Notemos que cualquier combinación con sentido de elementos de G_1 con $\circ$ y $\otimes$ puede expresarse como un diagrama de cables, el cual a su vez puede transformarse a un diagrama equivalente en posición general en virtud de la proposición 7. A su vez, todo diagrama en posición general puede ser expresado como un diagrama premonoidal, por lo tanto, cualquier morfismo en $\mathbf{MC}(G)$ se encuentra en $\mathbf{PMC}(G)$.
Por otro lado, la proposición 9 nos dice que cualquiera dos diagramas premonoidales tales que pueda deformarse, sin cruzar o cortar cables, son iguales. Pero de acuerdo al teorema de coherencia para el cálculo gráfico podemos concluir que $\mathbf{PMC}(G)/\sim$ satisface los axiomas de una categoría monoidal. □

4.2. Gramáticas monoidales

Veamos como podemos darle una interpretación lingüística a una signatura monoidal G. Podemos considerar los objetos de $\mathbf{MC}(G)$ como cadenas de símbolos de G_0, las cajas (o flechas generadores) de G_1 como reglas de producción y los morfismos en $\mathbf{MC}(G)$ como derivaciones. Formalmente podemos definir un lenguaje monoidal de la siguiente manera:

Definición 16. Una **gramática monoidal** es una signatura monoidal finita G de la siguiente forma

$$(V + B)^* \xleftarrow{\texttt{dom}} G \xrightarrow{\texttt{cod}} (V + B)^*$$

donde V es un conjunto de palabras llamada vocabulario y B es un conjunto de símbolos con un símbolo inicial distinguido $s \in B$ llamado el símbolo de oración.
Una oración $u \in V^*$ es gramaticalmente correcta si hay un morfismos $g : u \to s$ en $\mathbf{MC}$(G). Definimos el lenguaje generado por G como:

$$\mathcal{L}(G) = \{u \in V^* | \exists g : u \to s \text{ en } \mathbf{MC}(G)\}$$

A los lenguajes generados por gramáticas monoidales les llamamos lenguajes monoidales.

Ejemplo 8. Sabemos que los lenguajes libres de contextos son producidos por gramáticas de hipergráficas de la siguiente forma:

$$(V + B)^* \leftarrow G \rightarrow B$$

Ya que $B \subset (V + B)^*$, entonces son signaturas monoidales y, por lo tanto, es un lenguaje monoidal.

Tal como se prometió al inicio del capítulo, veremos que los lenguajes monoidales son exactamente los lenguajes recursivamente enumerables. Pero antes convendrá recordar las dos definiciones equivalentes, la primera en términos de las gramáticas que los generen y la segunda en términos del modelo de cómputo que los aceptan.

Definición 17. Un gramática de tipo 0 (o sin restricciones) es una tupla $G = (V, B, P, s)$ donde V es un vocabulario, B es un conjunto de símbolos no terminales con un símbolo distinguido s y P es un conjunto de reglas de producción de la forma $\alpha \to \beta$ donde $\alpha, \beta \in (V + B)^*$.
Si $\alpha \to \beta$ es una regla de producción, para cualquier $\gamma, \delta \in (V + B)^*$ definimos la derivación

$$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$$

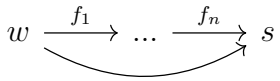
Definimos el lenguaje generado por G como:

$$\mathcal{L}(G) = \{v \in V^* | s \Rightarrow^* v\}$$

donde $\Rightarrow^*$ es la cerradura reflexiva y transitiva de $\Rightarrow$.

Teorema 4. *Los lenguajes monoidales son exactamente los generados por una gramática de tipo 0.*

Demostración. Primero veamos que todo lenguaje monoidal es producido por una gramática de tipo 0.
Sea G una gramática monoidal.
Para cada $f : u \to v$ en G_1 definimos la regla de producción $p_f : v \to u$ en P con $v, u \in (V + B)^*$. Notemos que si existen dos flechas generadoras $f, g : u \to v$, entonces $p_f = p_g$.
Sea $w \in \mathcal{L}(G)$, entonces existe $f : w \to s$ en $\mathbf{MC}(G)$, eso implica existen flechas generadoras tal que el siguiente diagrama conmuta



f

Entonces tenemos la siguiente derivación de P:

$$s \Rightarrow \cdots \Rightarrow w$$

Es decir, tenemos que $s \Rightarrow^* w$ y, por lo cual, w está en el lenguaje generado por la grámatica de tipo 0 (V, B, P, s). Por lo tanto, $\mathcal{L}(G) \subset \mathcal{L}(V, B, P, s)$. La otra contención se da leyendo el argumento anterior al revés y, por lo tanto, $\mathcal{L}(G) = \mathcal{L}(V, B, P, s)$. Resta probar que todo lenguaje generado por una gramática de tipo 0 es monoidal, aunque podríamos dar una demostración análoga, pospondremos su prueba. □

Como hemos mencionado anteriormente, cada nivel dentro de la jerarquía de Chosmky tiene asociado un modelo de cómputo capaz de reconocerlo, introduzcamos el modelo asociado a los lenguajes generados por una gramática de tipo 0.

Definición 18. Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ una máquina de Turing.
Definimos un movimiento de la siguiente manera: Dado $X_1X_2 \cdots X_{i-1}qX_i \cdots X_n$ en la banda y $\delta(q, X_i) = (p, Y, \leftarrow)$ tenemos que:

$$X_1X_2 \cdots X_{i-1}qX_i \cdots X_n \vdash_M X_1X_2 \cdots X_{i-2}pX_{i-1}Y \cdots X_n$$

Pero si $\delta(q, X_i) = (p, Y, \rightarrow)$, entonces

$$X_1X_2 \cdots X_{i-1}qX_i \cdots X_n \vdash_M X_1X_2 \cdots X_{i-1}YpX_{i+1} \cdots X_n$$

Definimos $\vdash_M^*$ como la cerradura reflexiva y transitiva de $\vdash_M$.
El lenguaje M aceptado por una máquina de Turing se define como

$$\mathcal{L}(M) = \{w \in \Sigma^* | q_0w \vdash_M^* \alpha s \beta \text{ con } s \in F \text{ y } \alpha \ , \ \beta \in \Gamma^*\}$$

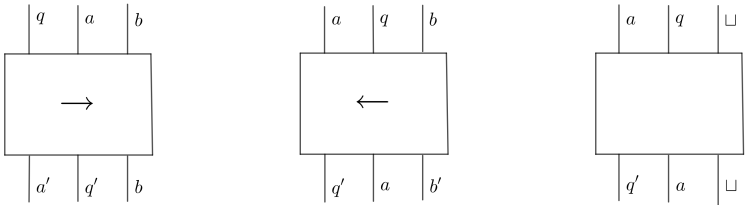
Es decir, el lenguaje producido por una máquina de Turing es el conjunto de cadenas del vocabulario tales que al inicializarse la banda con esas cadenas logramos alcanzar algún estado final. Ya estamos en condiciones de dar la siguiente definición:

Definición 19. Un lenguaje L es recursivamente enumerable si existe una máquina de Turing M que acepta L, es decir, $L = L(M)$.

Un resultado bien conocido es que que los lenguajes recursivamente enumerables tienen el mismo poder expresivo que los lenguajes generados por una gramática de tipo 0 [2]. Probemos que los lenguajes recursivamente enumerables son lenguajes monoidales.

Proposición 11. *Para todo lenguaje recursivamente enumerable L, existe una gramática monoidal G tal que $L = \mathcal{L}(G)$*

Demostración. Sea L un lenguaje recursivamente enumerable generado por la máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$.
Definimos $B = (\Gamma \setminus \Sigma) \cup Q$ y $V = \Sigma$ con $s \in B$ el símbolo de oración. Las reglas de producción son las siguientes:



donde $a, a', b, b' \in \Sigma$, $q, q' \in Q$ tales que $\delta(q, a) = (q', a', \rightarrow)$, $\delta(q, b) = (q', b', \leftarrow)$ y $\delta(q, \sqcup) = (q', \sqcup, \rightarrow)$. Además, agregamos las reglas de producción

$$x \rightarrow q_0x \qquad \text{y} \qquad xsy \rightarrow s$$

necesaria para inicializar la banda y para limpiarla.
Veamos que $L = \mathcal{L}(G)$
Sea $w \in L$. Entonces $q_0w \vdash_M^* \alpha s \beta$. Pero notemos que para cada aplicación $\vdash_M$ tenemos un morfismo que actúa de la misma manera, entonces tenemos morfismos tales que

$$q_0 w \xrightarrow{f_1} ... \xrightarrow{f_n} \alpha s \beta$$

Además como también tenemos las reglas de producción $w \to q_0 w$ y $\alpha s \beta \to s$ tenemos una derivación $w \to s$ en G. Por lo tanto, $w \in \mathcal{L}(G)$
Para el regreso, sea $w \in \mathcal{L}(G)$, entonces existe una derivación $w \to s$.
Observemos que por las reglas producción, tenemos la derivación es necesariamente de la siguiente forma

$$w \longrightarrow q_0 w \longrightarrow ... \longrightarrow \alpha s \beta \longrightarrow s$$

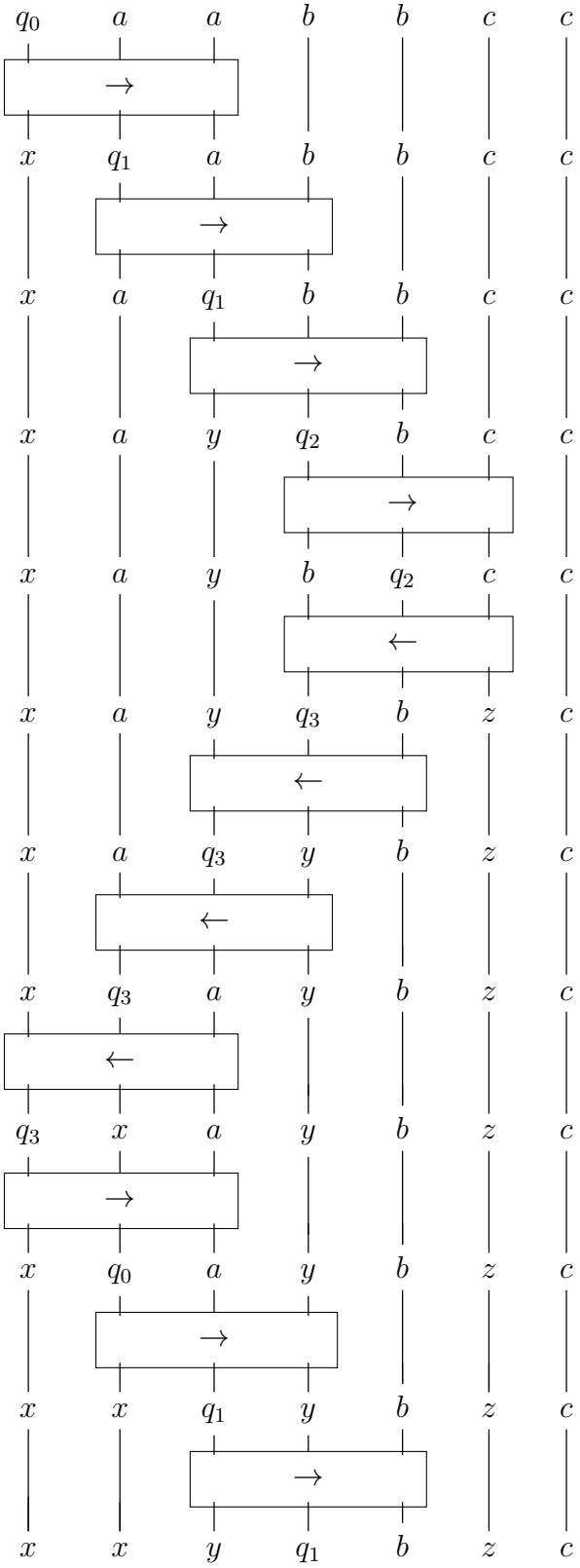
Pero la derivación intermedia solo puede lograrse con reglas de producción de la forma $\leftarrow$ y $\rightarrow$, las cuales corresponden con derivaciones en $\vdash_M$, por lo tanto, $q_0 w \vdash_M^* \alpha w \beta$.
Por lo tanto, $w \in L$.
Por lo tanto, $L = \mathcal{L}(G)$ □

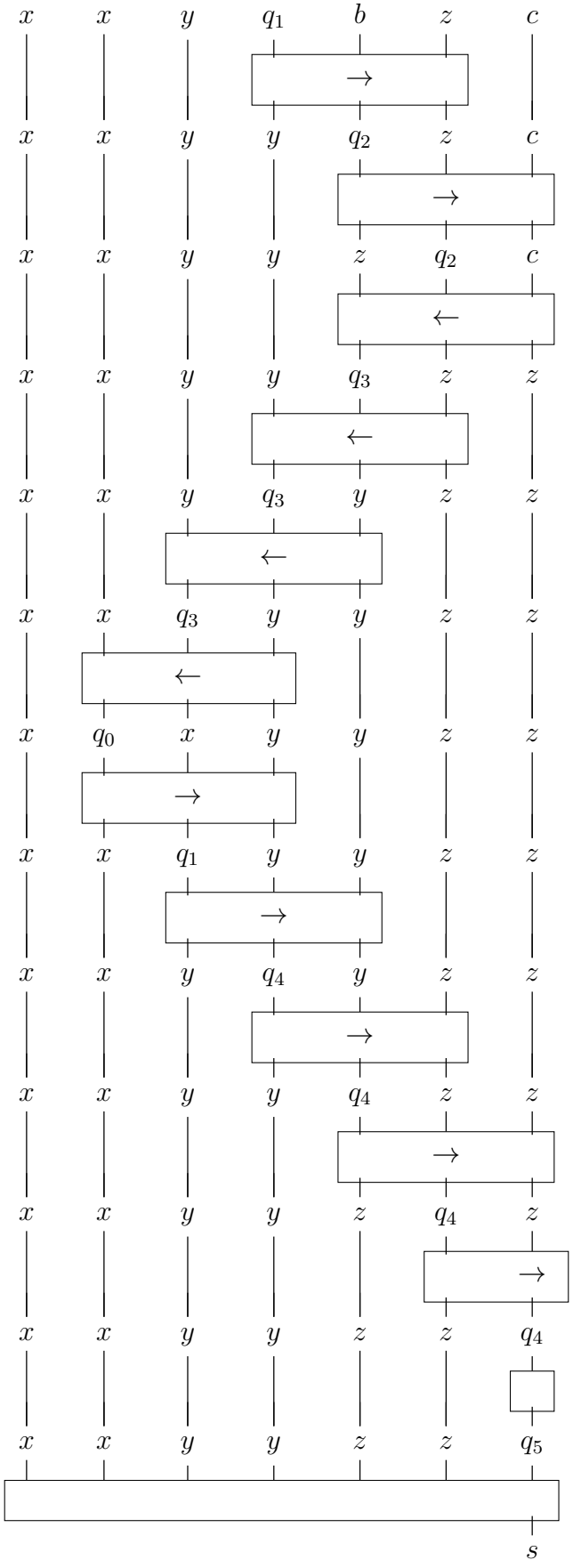
Terminemos este capítulo con el siguiente ejemplo

Ejemplo 9. El lenguaje $L = \{a^i b^i c^i | i \in \mathbb{N}\}$ no es regular, pero sí es recursivamente enumerable.
Una máquina de Turing que acepte el lenguaje anterior es la siguiente:

δ	a	b	c	x	y	z	$\sqcup$
q_0	$q_1, x, \rightarrow$				$q_4, y, \rightarrow$		$q_5, \sqcup, \rightarrow$
q_1	$q_1, a, \rightarrow$	$q_2, y, \rightarrow$			$q_1, y, \rightarrow$		
q_2		$q_2, b, \rightarrow$	$q_3, z, \leftarrow$			$q_2, z, \rightarrow$	
q_3	$q_3, a, \leftarrow$	$q_3, b, \leftarrow$	$q_3, c, \leftarrow$	$q_0, x, \rightarrow$	$q_3, y, \leftarrow$	$q_3, z, \leftarrow$	
q_4					$q_4, y, \rightarrow$	$q_4, z, \rightarrow$	$q_5, \sqcup, \rightarrow$
q_5							$q_5, s, \rightarrow$

Veamos la derivación en la gramática monoidal asociada de la palabra *aabbcc*


q_0 a a b b c c
$\rightarrow$
x q_1 a b b c c
$\rightarrow$
x a q_1 b b c c
$\rightarrow$
x a y q_2 b c c
$\rightarrow$
x a y b q_2 c c
$\leftarrow$
x a y q_3 b z c
$\leftarrow$
x a q_3 y b z c
$\leftarrow$
x q_3 a y b z c
$\leftarrow$
q_3 x a y b z c
$\rightarrow$
x q_0 a y b z c
$\rightarrow$
x x q_1 y b z c
$\rightarrow$
x x y q_1 b z c


x x y q1 b z c
→
x x y y q2 z c
→
x x y y z q2 c
←
x x y y q3 z z
←
x x y q3 y z z
←
x x q3 y y z z
←
x q0 x y y z z
→
x x q1 y y z z
→
x x y q4 y z z
→
x x y y q4 z z
→
x x y y z q4 z
→
x x y y z z q4
x x y y z z q5
s